VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Připojení USB zařízení k mikropočítači

USB Device Connection to Microcomputer

2014 Radim Žiška

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Radim Žiška**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Připojení USB zařízení k mikropočítači
USB Device Connection to Microcomputer

Zásady pro vypracování:

Vyberte z produkční řady firmy Microchip procesor, ke kterému bude možno připojit USB klávesnici. Připojení realizujte fyzicky i programově. Kromě klávesnice připojte k mikropočítači i LCD znakový displej a vytvořte prográmatorské rozhraní, kterým bude možné zajistit vstup z klávesnice a výstup na LCD displej.

1. Vyberte z produktové řady Microchip vhodné mikroprocesory, porovnejte jejich vlastnosti a dle dostupnosti na trhu vyberte vhodný typ.
2. Seznamte se s USB rozhraním a protokolem HID.
3. Vyberte vhodný LCD displej a navrhněte jeho přípojení k mikropočítači.
4. Navrhněte prototypové zapojení obvodů.
5. Napište potřebné programové vybavení, vytvořte ukázkovou aplikaci.
6. Vyhodnoťně spolehlivost navrženého řešení.

Seznam doporučené odborné literatury:

1. Datový list mikropočítače PIC24FJ64BG002, nebo jiného vhodného mikropočítače s OTG.
2. Datový list vybraného LCD.
3. Standard USB HID 1.1: http://www.usb.org/developers/devclass_docs/HID1_11.pdf.
4. Microchip USB Framework: http://www.microchip.com/usb.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Olivka**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. 5. 2014. .....

Abstrakt

Tato práce řeší problém připojení USB klávesnice k mikroprocesoru a zobrazení stisknutých kláves na LCD displej. Daný problém je řešen použitím aplikačních knihoven USB Framework, které jsou volně dostupné na stránkách firmy Microchip. Výsledkem práce je program, který umí zpracovat připojení USB klávesnice, zahájit komunikaci a data zaslaná z klávesnice zobrazovat na LCD displej. Práce neřeší odchylky v implementaci USB klávesnic mezí různými výrobci.

Klíčová slova

USB klávesnice, USB, HID, Microchip, PIC24FJ64GB002, LCD displej, MC16021E

Abstract

This thesis addresses the problem of connecting a USB keyboard to the microprocessor and display pressed keys on the LCD display. The problem is solved using USB Framework application libraries that are freely available on the Microchip website. The result is a program that can handle USB keyboard connection, initiation of communication and displaying data sent from USB keyboard on the LCD screen. The work does not address the differences in the implementation of USB keyboard between different manufacturers.

Keywords

USB keyboard, USB, HID, Microchip, PIC24FJ64GB002, LCD display, MC16021E

Seznam použitých zkratek a symbolů

HW – Hardware

OEM – Original Equipment Manufacturer

LCD – Liquid Crystal Display

HID – Human Interface Device

USB – Universal Serial Bus

VŠB - TUO – Vysoká Škola Báňská - Technická Univerzita Ostrava

ASCII – American Standard Code for Information Interchange

Obsah

1 Úvod

Jako téma mé bakalářské práce jsem si vybral problematiku připojení USB klávesnice k mikroprocesoru, kde konečným výsledek je zobrazování stisknuté klávesy na LCD displej. Řešení jsem rozdělil na několik kapitol, kde se daným problémům věnuji nejdříve z teoretické stránky a poté implementační.

V kapitole 2 se dle zadání nejdříve věnuji výběru hardwarových a softwarových komponent. Definuji v ní požadavky a následně popisuji vlastnosti vybraných komponent.

V kapitole 3 uvádím implementační rozhodnutí, zapojení a konečné ovládání LCD.

V kapitole 4 popisuji USB architekturu, probírám HID protokol a rozebírám strukturu předávaných dat od USB zařízení.

V kapitole 5 se zabývám konfigurací mikroprocesoru, aby vystupoval jako USB host zařízení. Dále probírám stavový automat USB zařízení, kterým musí program projít přesně dle definice, aby zapojení fungovalo.

V kapitole 6 využívám zjištěných informací z kapitoly 4, 5 a provádím implementaci za pomoci aplikačních knihoven firmy Microchip. K vzniklému programu obsluhující USB zařízení přidávám můj program obsluhující LCD displej a vytvářím konečný program pro mikroprocesor dle zadání.

V kapitole 7 shrnuji problémy během vývoje a výslednou spolehlivost řešení.

2 Výběr hardwarových a softwarových komponent

Prvním krokem je výběr všech potřebných součástí práce, které budu potřebovat k vývoji po celou dobu. Začnu výběrem hardwarových komponent, jmenovitě mikroprocesor, LCD a další potřebné příslušenství kolem. Pokračovat budu výběrem potřebného softwaru pro vývoj aplikací a vytvoření schémat zapojení.

2.1 Výběr z produktové řady Microchip vhodného mikroprocesoru

Zde se budu zabývat výběrem konkrétního mikroprocesoru. Všechny požadavky splňuje velké množství mikroprocesorů z řady Microchip, proto budu specifikovat požadavky. Tento velký výběr rozdělím na výběr série a poté konkrétního mikroprocesoru. Kritéria splňují i jiní výrobci, ale výběr probíhá jen z řady Microchip. Důvodů je několik, v rámci studia na VŠB-TUO jsem se seznámil s vývojem na Microchip procesorech a také s ovládáním vývojového prostředí. Díky tomu jsou lehce dostupné programátory a odborná rada při řešení problému při vývoji.

2.1.1 Požadavky

Abych zredukoval tento výběr, přidal jsem následující požadavky. Potřebuju, aby mikroprocesor dobře seděl do kontaktního nepájivého pole, to znamená, že potřebuju pouzdro typu SPDIP. Dalším požadavkem je co nejnižší řada s co možná nejnižším počtem pinů. Po tomto užším výběr jsem se dostal k sérii procesorů PIC24FJ. Z této série všem požadavkům vyhovují pouze tyto dva procesory PIC24FJ32GB002, PIC24FJ64GB002. Procesory jsem začal porovnávat a z vlastností vyplynulo, že mohou být vhodné oba. Procesory se od sebe liší jen velikostí programové paměti, pro PIC24FJ32GB002 je tato velikost 32KB a pro PIC24FJ64GB002 to je 64KB. Ověřím si tedy, jestli je 28 pinů dostačující, z toho je 21 pinů určeno na vstup/výstup. Pro připojení USB se využiji 2 specifické piny, pro LCD 7 nebo 11 pinů, kde počet je závislý na implementaci a pro programátor další 2. Celkový počet je tedy 11 nebo 15 pinů. Dostupný počet vhodných pinů pro toto využití je 19. Proto nic nebrání zvolení jednoho ze dvou kandidátů, protože nevím, jak rozsáhlý bude program a vycházet budu z aplikační knihovny od Microchip, ve které jsou knihovny psané tak, aby podporovaly široká rozsah mikroprocesorů, volím PIC24FJ64GB002. Z pohledu vlastností splňuje všechny požadavky a je dobře dostupný na trhu za cenu přibližně 150Kč. Pokud by byl třeba program a zapojení rozšířit, tak lze použít procesor z rodiny PIC24FJ64GB004.

2.1.2 PIC24FJ64GB002

Jedná se o mikroprocesor založený na 16bitové architektuře. Obsahuje interní oscilátory o frekvenci 32kHz a 8MHz s 4X PLL obvody pro až 32MHz. Podporuje USB v2.0 s On-the-Go funkcionalitou, kterou ale v práci nebudu využívat. V USB zapojení umí vystupovat jako host i device. Celkový počet pinů je 28, z nichž je 21 určeno pro vstup/výstup.

2.2 Výběr vhodného LCD displeje

Před výběrem vycházím z toho, že hlavním kriteriem je dostupnost, protože na displej není kladen žádný výkonnostní požadavek a většina pro mě zajímavých vlastností u displeje je stejná. Jako dostačující bude dvouřádkový displej, s počtem znaků 16 na řádek. Jeden takový dostupný displej je MC16021E-TGR.

2.2.1 MC16021E-TGR

Jedná se o dvouřádkový displej s maximem 16 znaků na řádek s nastavitelným kontrastem. Rozlišení pro jeden znak je 5x7 bodů. Potřebné napájení pro tento displej je 5V. Celkový počet pinů je 14, z toho 8 je datových. Podle implementace datové komunikace využiji všech 8 nebo 4.

2.2.2 Popis pinů

VSS – zemnění

VDD – napájení 5V

VEE – regulace napájení pro LCD kontrast

RS – výběr registru: 0 – instrukční, 1 – datový

R/W – čtení/zápis: 0 – zápis, 1 – čtení

E – příznak pro zahájení čtení/zápisu

D0 až D7 – datové bity

2.3 Další hardwarové příslušenství

Pro zapojení potřebuji ještě další komponenty. Hlavní z nich je kontaktní nepájivé pole, dostačující je PROSKIT BX-4112N (840 pinů). Jako napájení je zapotřebí zdroj 5V pro USB a LCD část obvodu a 3,5V pro mikroprocesor. Tento problém jsem vyřešil zapojením jednoho 5V zdroje a snížením napětí na 3,5V pomocí dvou diod 1N4148. K programování mikroprocesoru potřebuji programátor, protože během studia jsem se seznámil a využíval jsem Microchip MPLAB ICD 2, tak jsem ověřil, jestli tento programátor podporuje mikroprocesor PIC24FJ64GB002. Zjistil jsem, že podporuje a jelikož je využíván při výuce, byl také dostupný k zapůjčení. S tímto výběrem souvisí výběr programového vybavení.

2.4 Výběr softwarového vybavení

Spolu s výběrem programátoru je svázaná volba softwaru pro vývoj. S ICD 2 je vhodné použít MPLAB IDE. Konkrétně jsem zvolil nejnovější dostupnou verzi podporující programátor ICD 2, tedy MPLAB IDE v8.89. Dostupná je i novější verze MPLAB X IDE, ale zde už není podporován

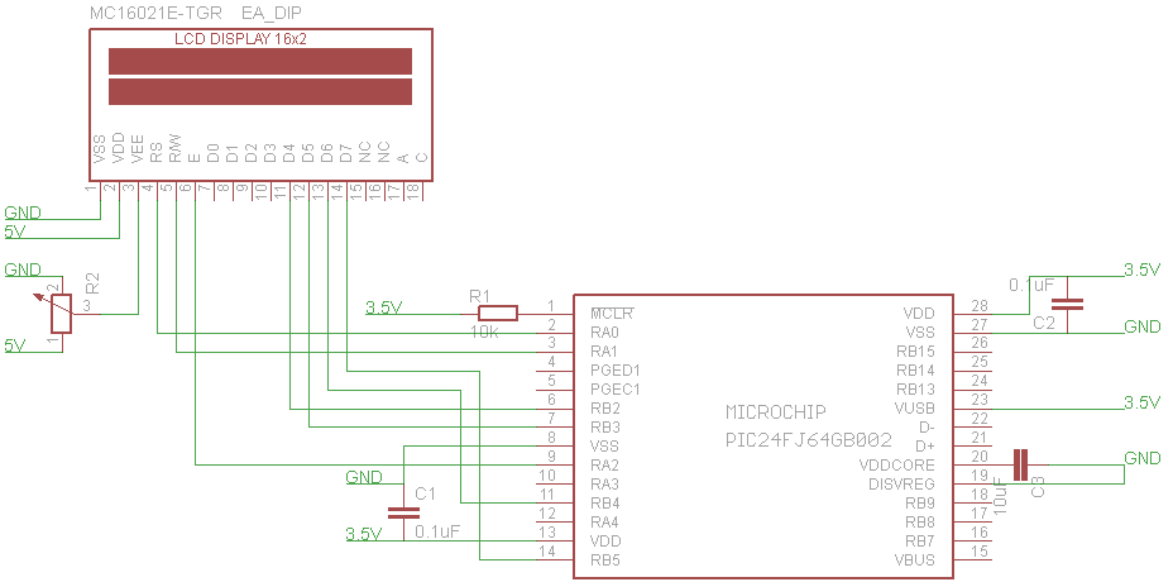
programátor ICD 2. Jako programovací jazyk je dán jazyk C. K tomuto prostředí a programovacímu jazyku jsou dostupné různé kompilátory. Vhodné kompilátory k programování mikroprocesorů PIC24 jsou C30 a XC16. Po odzkoušení obou jsem zvolil kompilátor C30 verze 3.31, který optimalizoval výstup, ale jen po zkušební dobu.

Jako software k tvorbě technických schémat zapojení mi byl doporučen program Eagle, po odzkoušení jsem ho použil na tvorbu všech schémat pro práci.

3 Zapojení a ovládání LCD displeje

Před zapojením LCD jsem se rozhodl, že budu implementovat 4 bitovou komunikaci mezi LCD a mikroprocesorem. Není tedy povinnost zapojovat datové piny D0 až D3. Komunikace probíhá pouze na pinech D4 až D7 a ovládací signály jsou dané na pinech RS, R/W a E. Mimo napájeni na pinech V_{SS} a V_{DD} mám ještě zapojený potenciometr na pinu V_{EE}, pomocí kterého lze regulovat kontrast na displeji.

3.1 Schéma zapojení



Obrázek 1: Schéma zapojení LCD

3.2 Popis komunikace

Protože komunikace probíhá pouze na 4 bitech, musí se celé 8bitové slovo posílat po částech. První část jsou horní bity 7-4 a druhá spodní bity 3-0. Abych mohl využívat tento způsob komunikace, musím provést 4 bitovou inicializaci dle dokumentace výrobce. Pro můj displej vypadá inicializační funkce následovně.

```
void LcdInit(void)
{
      LCD_RS = 0;         // výběr registu instrukcí
      __builtin_nop();    // čekání na stabilizaci výstupu na portu
      LCD_RW = 0;         // příznak zápisu

      DelayMs(40);        // čekání na stabilizaci napětí na displeji

      SendPortB(0x03);    // inicializační instrukce
```

```
    EnableSet();      // příznak pro zpracování instrukce
    DelayMs(10);      // čekání před další instrukcí

    SendPortB(0x03);  // inicializační instrukce
    EnableSet();
    DelayMs(5);

    SendPortB(0x03);  // inicializační instrukce
    EnableSet();
    DelayMs(5);

    SendPortB(0x02);  // inicializační instrukce
    EnableSet();
    DelayMs(5);

    LcdCmnd(0x28);    // nastavení 4bit módu, 5x7 fontu a 2 řádky
    LcdCmnd (0x08);   // vypnutí zobrazování na displej
    LcdCmnd (0x01);   // reset dispeje
    DelayMs(2);       // čekání před další instrukcí
    LcdCmnd (0x06);   // pohyb z leva do prava po displeji
    LcdCmnd (0x0D);   // zapnutí displeje, blikání kurzoru na displeji
}
```

Ukázka z kódu 1: LCD inicializace

3.3 Funkce pro ovládání displeje

K ovládání displeje používám tuto sadu funkcí, kterou jsem naprogramoval pro pohodlné použití později při zobrazování informací na displej:

```
void LcdClear(void);                // vymaže zobrazovaný obsah a kurzor
                                se
                                    // vrátí na začátek prvního řádku

void LcdPuts(const char * s);       // vypíše řetězec s od pozice kurzoru

void LcdPutch(unsigned char c);     // vypíše znak c na pozice kurzoru

void LcdGoToLine1();                // přesun kurzoru na první řádek

void LcdGoToLine2();                // přesun kurzoru na druhý řádek

void LcdInit(void);                 // inicializace displeje
```

4 USB

USB (Universal Serial Bus) je univerzální sériová sběrnice. Dnes velmi využívaná pro připojení různých druhů periferních zařízení. Existuje několik standardů a v mojí práci se zabývám verzí USB 1.1 (Low Speed).

Přenos dat je řešen na linkové vrstvě pomocí rámců, kde jednotlivé bity jsou kódovány metodou NRZI (Non Return to Zero Inverted).

4.1 Architektura USB

Při komunikaci po této sběrnici existuje pouze jedno zařízení typu host. To platí, i případě využití různých rozbočovačů. Zbytek připojených zařízeni jsou typu device. Počet vrstev rozbočovačů je limitován, ale v mojí práci se zabývám komunikací jeden host a jeden device, kde device je HID zařízení, konkrétně klávesnice.

4.1.1 USB host

Musí detekovat připojení a odpojení zařízení, spravovat komunikaci s připojeným zařízením, ukládat si stav a aktivitu a dodávat napájení. Bývá implementován jako kombinace hardware, firmware nebo software. Musí implementovat ovládací prvky dle podporovaných zařízení.

4.1.2 USB device

Jsou všechny ostatní zařízení připojené do USB systému. Dělí se na různé třídy, např. rozbočovač, tiskárna, datové úložiště, HID, atd. Všechny device zařízení musí definovat sebe identifikaci, obecnou konfiguraci a dodržovat konzistentní zásady chování definované USB stavovaným automatem.

4.2 Třída HID

Human Interface Device je jedna z tříd USB zařízení s kódovým označením *bInterfaceClass* = 3. HID třída definuje zařízení až v interface deskriptoru. Do HID třídy patří převážně zařízení, pomocí kterých uživatel ovládá daný systém. Mezi hlavní zařízení se řadí myš, klávesnice, ovládací panely, atd. Tyto zařízení zapadají do podtříd. Já se zaměřím pouze na podtřídu klávesnice.

4.2.1 Podtřída klávesnice

Podtřída klávesnice má kódové označení *bInterfaceSubClass* = 1 a *bInterfaceProtocol* = 1, kde *bInterfaceSubClass* znamená, že zařízení podporuje boot interface, pokud není nastaveno na 1 je *bInterfaceProtocol* ignorován, *bInterfaceProtocol* udává, o jaké zařízení se jedná, v rámci mojí práce se zabývám jen klávesnicí.

4.2.2 Komunikace HID zařízení

HID zařízení využívá pro komunikaci dva kanály. Výchozí ovládací (endpoint 0) kanál, který musí být definován v každém zařízení a kanál pro přerušení.

Výchozí kanál se používá pro:

- Obdržení a odpovědi na požadavky týkající se USB ovládacích a dat třídy.
- Odesílání dat při periodickém dotazování host zařízením
- Obdržení dat od host zařízení

Kanál přerušení se používá pro (z pohledu host zařízení):

- Obdržení dat od device zařízení
- Odeslání dat do device s co nejmenším zpožděním

Komunikace pomocí přerušení je povinná pouze ve směru od device do host. Pokud opačný směr není v deskriptorech definován, tak se používá výchozí kanál, pro odesílaní dat.

4.3 Deskriptory USB zařízení

Jedna z prvních věcí, kterou musí host zařízení vykonat, je získání a zpracování deskriptorů zařízení. Deskriptor je datová struktura přesně definovaného formátu. Každý deskriptor začíná bajtem, který obsahuje celkový počet bajtů v deskriptoru a následuje další bajt, který identifikuje typ deskriptoru.

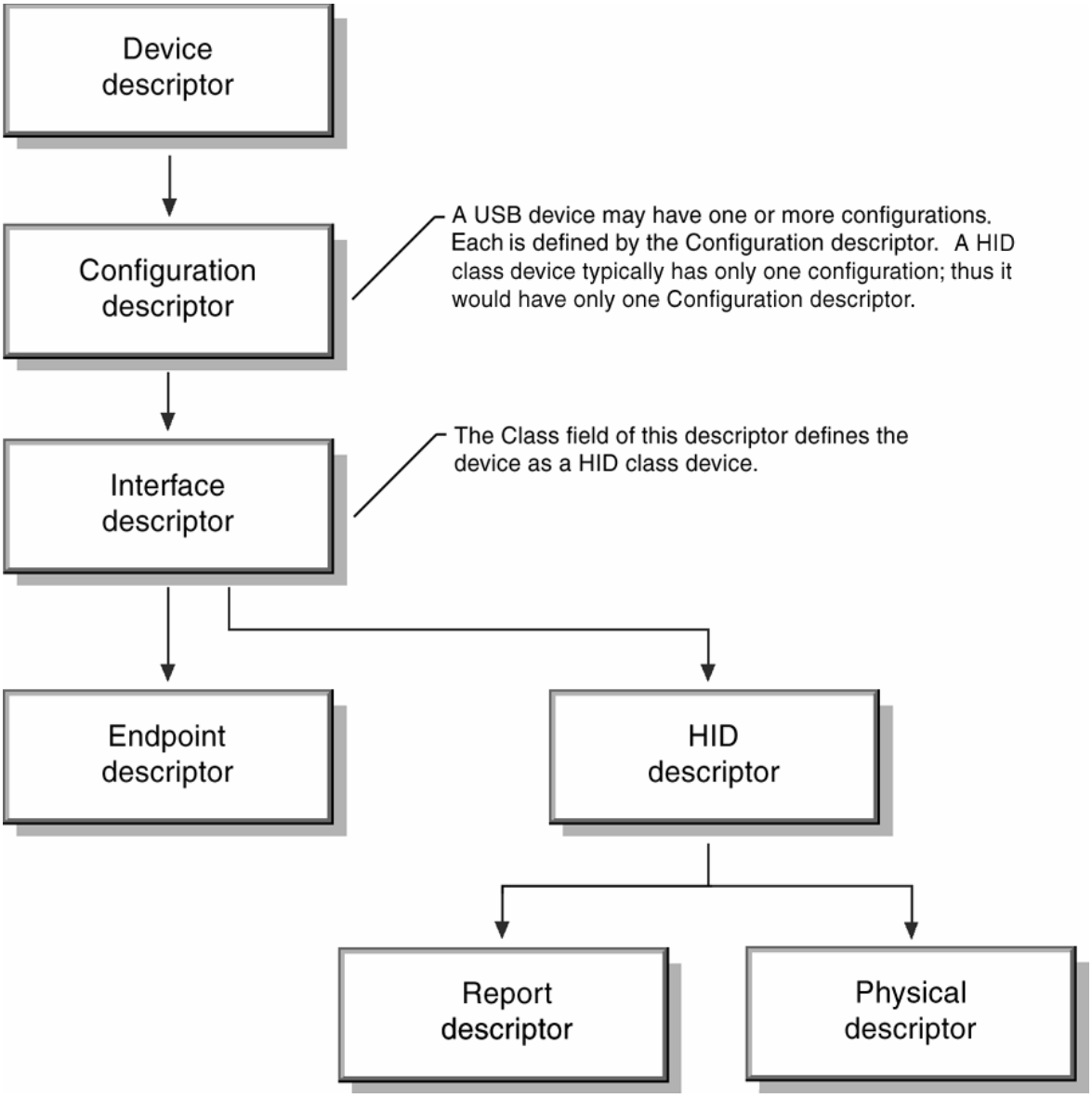
Použití deskriptorů umožňuje stručné ukládání atributů jednotlivých konfigurací, protože každá konfigurace může znovu použít deskriptor nebo části deskriptorů z jiných konfigurací, které mají stejné vlastnosti.

Tam, kde je to vhodné, popisovače obsahují odkazy na řetězcové deskriptory, které poskytují zobrazitelné informace, popisující deskriptor ve formě čitelné pro člověka. Zahrnutí řetězcových deskriptorů je volitelné. Nicméně referenční pole v rámci deskriptorů jsou povinné. Pokud zařízení nepodporuje řetězcové deskriptory, řetězce referenčních polí musí být nastaveny na nulu, aby naznačily, že řetězcové deskriptory nejsou k dispozici.

Pokud se vrací deskriptor s hodnotou své délky, která je menší, než je definováno ve specifikaci USB, deskriptor je neplatný a měl by být odmítnut host zařízením. Pokud se vrací deskriptor s hodnotou své délky, která je větší než definovaná ve specifikaci USB, extra bajty jsou ignorovány host zařízením, ale následující deskriptor je umístěn na místě daném předchozí délkou.

4.3.1 Struktura HID device deskriptoru

U HID zařízení je výjimka, že DeviceClass, DeviceSubClass a DeviceProtocol se definují až v interface deskriptoru a mají vlastní definovaný význam, viz výše v podkapitole 3.2.1. Na obrázku 2 je zobrazena struktura po sobě jdoucích deskriptorů [2].



Obrázek 2: Struktura device deskriptoru

4.3.2 Device deskriptor

Popisuje obecné informace o zařízení. Obsahuje obecně platné informace pro všechny konfigurace. Device deskriptor je pouze jeden.

Počáteční bajt	Název atributu	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bajtech
1	bDescriptorType	1	Constant	Typ device deskriptor
2	bcdUSB	2	BCD	Verze USB specifikace
4	bDeviceClass	1	Class	Třída zařízení
5	bDeviceSubClass	1	SubClass	Podtřída zařízení
6	bDeviceProtocol	1	Protocol	Protokol zařízení
7	bMaxPacketSize0	1	Number	Maxmální velikost packetu pro endpoint 0
8	idVendor	2	ID	Id výrobce
10	idProduct	2	ID	Id produktu
12	bcdDevice	2	BCD	Číslo vydané řady
14	iManufacturer	1	Index	Index řetězce popisující výrobce
15	iProduct	1	Index	Index řetězce popisující produkt
16	iSerialNumber	1	Index	Index řetězce popisující sériové čislo
17	bNumConfigurations	1	Number	Počet možných konfigurací

Tabulka 1: Standardní device deskriptor

4.3.3 Konfigurační deskriptor

Popisuje informace o dané konfiguraci zařízení a kolik rozhraní podporuje. Pokud host požaduje konfigurační deskriptor, tak všechny navazující interface a enpoint deskriptory jsou vráceny.

Počáteční bajt	**Název atributu**	**Velikost**	**Hodnota**	**Popis**
0	bLength	1	Number	Velikost deskriptoru v bajtech
1	bDescriptorType	1	Constant	Typ konfigurační deskriptor
2	wTotalLength	2	Number	Celková délka dat konfigurace pro všechny deskriptory
4	bNumInterfaces	1	Number	Počet rozhraní
5	bConfigurationValue	1	Number	Argument pro výběr této konfigurace
6	iConfiguration	1	Index	Index řetězce popisující konfiguraci
7	bmAttributes	1	Bitmap	Konfigurační charakteristiky
8	bMaxPower	1	mA	Max odběr proudu, vyjádřeno v 2mA(tzn. 50 = 100mA)

Tabulka 2: Standardní konfigurační deskriptor

4.3.4 Interface deskriptor

Popisuje konkrétní rozhraní pro konfiguraci. Konfigurace poskytuje jedno nebo více rozhraní, každé s žádným nebo více endpoint deskriptory popisující unikátní sadu endpointů. Pokud rozhraní používá pouze endpoint 0, tak nenásleduje endpoint deskriptor a bNumEndpoints je rovno 0.

Počáteční bajt	Název atributu	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bajtech
1	bDescriptorType	1	Constant	Typ interface deskriptor
2	bInterfaceNumber	1	Number	Index rozhraní
3	bAlternateSetting	1	Number	Hodnota pro výběr alternativního nastavení
4	bNumEndpoints	1	Number	Počet využitých endpointů
5	bInterfaceClass	1	Class	Kód třídy
6	bInterfaceSubClass	1	SubClass	Kód podtřídy
7	bInterfaceProtocol	1	Protocol	Kód protokolu
8	iInterface	1	Index	Index řetězce popisující rozhraní

Tabulka 3: Standardní interface deskriptor

4.3.5 Endpoint deskriptor

Každý endpoint použitý v rozhraní má svůj vlastní deskriptor. Endpoint 0 nemá žádný deskriptor. Enpoint informace jsou důležité pro host zařízení k správnému datovému přenosu.

Počáteční bajt	Název atributu	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bajtech
1	bDescriptorType	1	Constant	Typ endpoint deskriptor
2	bEndpointAddress	1	Endpoint	Kódována Adresa endpointu
3	bmAttributes	1	Bitmap	Atributy endpointu
4	wMaxPacketSize	2	Nuimber	Maximální velikost packetu
6	interval	1	Number	Interval periodického dotazování

Tabulka 4: Standardní endpoint deskriptor

4.3.6 HID deskriptor

Identifikuje délky a typy podřízených deskriptorů zařízení.

Počáteční bajt	Název atributu	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bajtech
1	bDescriptorType	1	Constant	Typ HID deskriptor
2	bcdHID	2	Number	Verze HID specifikace
4	bCountryCode	1	Number	Kód země lokalizovaného HW
5	bNumDescriptors	1	Number	Počet deskriptorů tříd
6	bDescriptorType	1	Constant	Identifikace třídy deskriptoru
7	wDescriptorLength	2	Number	Délka deskriptoru
9	[bDescriptorType]	1	Constant	Identifikace třídy volitelného deskriptoru
10	[wDescriptorLength]	2	Number	Délka volitelného deskriptoru

Tabulka 5: Standardní HID deskriptor

4.3.7 Report deskriptor

Tento deskriptor není jako ostatní deskriptory, nelze ho obecně popsat jednoduchou tabulkou hodnot. Délka a obsah reportu se liší dle zařízení a jeho požadavků. Protože se v mojí práci zabývám jenom klávesnicí, tak můžu rozepsat konkrétní použité reporty.

Report stisku na klávesnici:

Standardní USB Klávesnice posílá 8 bajtu dlouhý report do host zařízení.

Bajt	Popis
0	Modifikující klávesy
1	Rezervováno
2	Kód klávesy 1
3	Kód klávesy 2
4	Kód klávesy 3
5	Kód klávesy 4
6	Kód klávesy 5
7	Kód klávesy 6

Tabulka 6: Input report

Některé klávesnice mohou obsahovat vlastní implementaci, která se neshoduje se standardní implementaci. Takové klávesnice musí poskytnout vlastní tabulku znaků, aby bylo možné kódy kláves přiřadit ke správným klávesám. Bajt 1 je rezervován pro OEM.

Report pro klávesnici:

Tímto reportem nastavuje host zařízení stav LED diod. Délka reportu je jeden bajt.

Bit	**Popis**
0	Num Lock
1	Caps Lock
2	Scroll Lock
3	Compose
4	KANA
5 až 7	Konstanta

Tabulka 6: Output report

Standardní klávesnice má tři LED diody k zobrazení stavu Num Lock, Caps Lock a Scroll Lock. Report musí obsahovat stavy všech diod. Stav logická 0 je vypnuto a stav logická 1 zapnuto.

4.3.8 Physical deskriptor

Popisují, kterou částí těla se daný kontrolní prvek ovládá, jedná se o dobrovolnou položku.

5 Připojení USB k mikroprocesoru

Pro připojení USB konektoru se používají předem definované piny na mikroprocesoru. Pro moje zapojení používám konektor typu A. Konektor typu A obsahuje 4 piny. Jmenovitě +5V, GND, D+ a D-. Konektory D+ a D- jsou datové piny, také označováno jako diferenciální pár.

5.1 Konfigurace mikroprocesoru

Každý program musí obsahovat konfiguraci. Pro obsluhu USB musí být speciálně nastaveny děličky pro oscilátor, aby procesor měl přesně danou frekvenci. Tato konečná frekvence musí být 48MHz. K dispozici mám interní oscilátor, který mám nastavený na frekvenci 4MHz. Konečnou frekvenci dostanu díky PLL obvodu, který ze vstupních 4MHz udělá 96MHz výstup. Aby šlo dosáhnout vstupní frekvence 4MHz z různých zdrojů je před PLL obvodem zabudovaná nastavitelná předdělička. Jakmile máme 96MHz je tato frekvence dělena na 48MHz pro USB a 32MHz pro systémové hodiny. Frekvenci 32MHz lze vydělit pomocí zabudované nastavitelné děličky.

```
_CONFIG1(WDTPS_PS1 & FWPSA_PR32 & WINDIS_OFF & FWDTEN_OFF & ICS_PGx1 &
GWRP_OFF & GCP_OFF & JTAGEN_OFF)

_CONFIG2(POSCMOD_NONE & I2C1SEL_PRI & IOL1WAY_OFF & OSCIOFNC_ON &
FCKSM_CSDCMD & FNOSC_FRCPLL & PLL96MHZ_ON & PLLDIV_NODIV & IESO_OFF)

_CONFIG3(WPFP_WPFP0 & SOSCSEL_IO & WUTSEL_FST & WPDIS_WPDIS &
WPCFG_WPCFGDIS & WPEND_WPENDMEM)

_CONFIG4(DSWDTPS_DSWDTPS3 & DSWDTOSC_LPRC & RTCOSC_LPRC & DSBOREN_OFF &
DSWDTEN_OFF)
```

Ukázka z kódu 2: Konfigurace mikroprocesoru

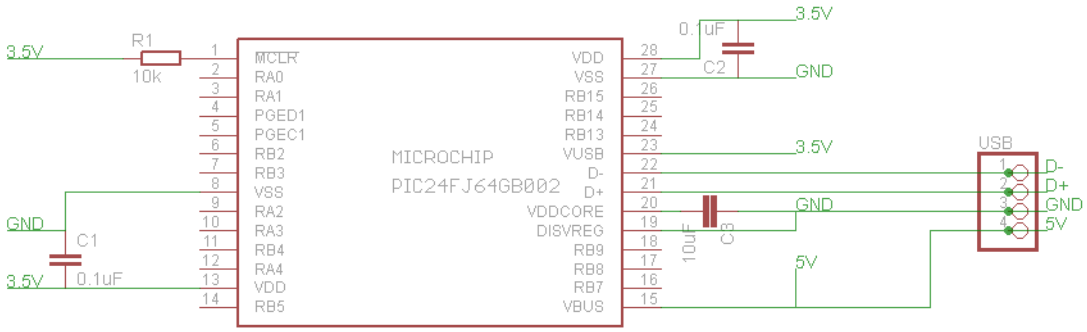
V prvním konfiguračním slově mě zajímá hlavně nastavení ICS_PGx1, kterým určuji, kde je zapojený programátor ICD 2. Ostatní funkce jsou vypnuty.

V druhém konfiguračním slově nastavuji oscilátor. FNOSC_FRCPLL určuje, že bude použit interní oscilátor napojený na PLL obvod. PLL96MHZ_ON zapíná PLL obvod na zvýšení interní frekvence pro USB obvod. OSCIOFNC_ON zapne pin RA3 a můžu ho využít jako vstup/výstup. Ostatní funkce jsou vypnuty.

V třetím konfiguračním slově nastavuji ochrany proti zápisu a pomocí SOSCSEL_IO nastavuji, aby piny RA4 a RB4 měly vstup/výstupní funkci.

Ve čtvrtém konfiguračním slově se nastavuje funkcionalita watchdog a uspávání, tyto funkce nepoužívám a jsou vypnuty.

5.2 Schéma zapojení



Obrázek 3: Schéma zapojení USB

5.3 Popis komunikace

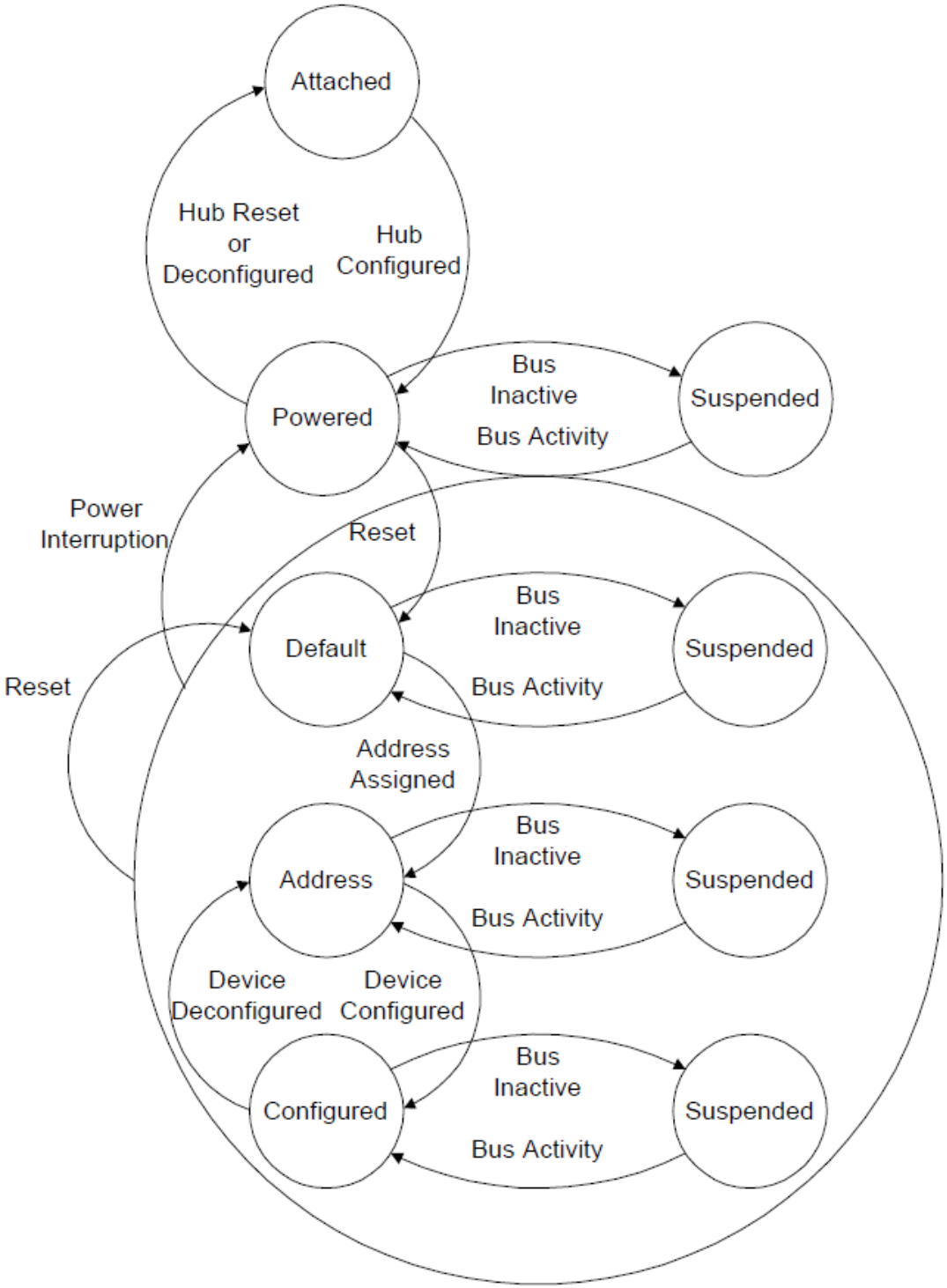
Při implementaci USB komunikace, bude můj mikroprocesor vystupovat v roli host a připojená klávesnice v roli device. Device zařízení musí dodávat informace o sobě ve formě deskriptorů. Některá zařízení můžou využívat OTG to znamená, že dokážou dynamicky měnit svou roli z device na host a naopak.

5.3.1 Popis zahájení komunikace

Před zahájením zpracování deskriptoru musí host zařízení zjistit, zda se připojilo zařízení a inicializovat základní komunikaci. V případě klávesnice zahájí USB komunikaci verze 1.1, neboli Low-Speed s přenosovou rychlostí 1,5 Mbit/s. Klávesnice to oznámí po připojení tím, že na pinu D- nastaví logickou 1, na základě této informace host zřízení bude používat Low-Speed. Spolu s nastavením D- mikroprocesor vyvolá přerušení. V tomto přerušení si host ověří, jakou rychlost bude používat a pošle do USB zařízení příznak pro reset. Po korektním resetu host dále vykonává enumeraci dle definice USB specifikací.

Dle obrázku 3 zařízení prochází stavy [1]:

- Attached – Připojeno
- Powered – Napájeno
- Default – Výchozí stav
- Address – Adresováno
- Configured – Nakonfigurováno
- Suspended – Neaktivní



Obrázek 4: Stavový diagram

Enumerace obnáší tyto kroky:

Port, ke kterému je device zařízení připojeno informuje host zařízení o této události pomocí odpovědi o změně statusu kanálu (více v sekci 11.12.3 USB 2.0 specifikací). V tomto momentu je USB zařízení napájené a port, na který je připojené deaktivován.

Host zařízení zjistí, o jakou konkrétní změnu šlo dotazováním. Tento krok se týká při připojení na hub, kde host potřebuje dodatečné informace.

Nyní host zařízení ví, na který port se připojilo zařízení a čeká alespoň 100ms kvůli dokončení procesu připojení a stabilizaci napětí. Následně aktivuje port a vydá příkaz k restartu zařízení (časování resetu v sekci 7.1.7.5 USB specifikací).

Na daném portu se vykoná reset zařízení (více v sekci 11.5.1.5 USB specifikací). Po dokončení resetu, je port aktivní. USB zařízení je nyní ve výchozím stavu a nemůže odebírat více jak 100mA ze zdroje napájení. Všechny stavy a registry jsou ve výchozím stavu a zařízení odpovídá na výchozí adrese.

Host zařízení přiřadí unikátní adresu pro zařízení a zařízení se tímto dostává do adresovacího stavu.

Předtím, než zařízení obdrží unikátní adresu, je výchozí kanál dostupný na výchozí adrese. Host zařízení z deskriptoru zjistí maximální datovou zátěž, kterou výchozí kanál může použít.

Host zařízení zpracuje konfigurační informace ze zařízení přečtením konfigurací od 0 po *n*-1, kde *n* je počet konfigurací. Tento proces může trvat několik milisekund.

Dle informací z konfigurace a jak bude zařízení použito, host zařízení přiřadí konfigurační hodnotu k zařízení. Zařízení je nyní v nakonfigurovaném stavu a všechny koncové body v konfiguraci mají svou popsanou charakteristiku. USB zařízení může nyní odebírat V_{BUS} napájení popsané ve svém deskriptoru pro danou konfiguraci. Z pohledu zařízení, je připraveno k použití.

5.3.2 Implementace zahájení komunikace

Celá implementace vychází z programové knihovny Microchip solutions vydané 15. června 2013, konkrétní ukázková aplikace, kterou jsem upravil dle požadavků, se jmenuje „Host - HID - Keyboard“ [7]. Program bez připojeného zařízení hledá cyklicky, jestli se změnil stav a bylo připojeno zařízení. Ukázka z funkce `main()` po inicializaci:

```
while(1)
{
    USBTasks();
    App_Detect_Device();
}
```

Ukázka z kódu 3: Jádro

Registr obsahující informace o přerušení je U1IR a registr povolující daný typ přerušení je U1IE. Pokud je dané přerušení povoleno, reprezentující bit v U1IE je nastaven na log 1. Pokud nastane přerušení je reprezentující bit v U1IR nastaven na log 1. Obsah obou registrů je následující [5]:

STALLIF	ATTACHIF	RESUMEIF	IDLEIF	TRNIF	SOFIF	UERRIF	DETACHIF
Bit 7							Bit 0

Tabulka 2: U1IR v host módu

Bit 15-8 Neimplementováno

Bit 7 STALLIF: Handshake

Bit 6 ATTACHIF: Zařízení připojeno

Bit 5 RESUMEIF: Obnovení

Bit 4 IDLEIF: Neaktivita

Bit 3 TRNIF: Zpracování tokenu dokončeno

Bit 2 SOFIF: Start-of-Frame token

Bit 1 UERRIF: USB Error

Bit 0 DETACHIF: Zařízení odpojeno

STALLIE	ATTACHIE	RESUMEIE	IDLEIE	TRNIE	SOFIE	UERRIE	DETACHIE
Bit 7							Bit 0

Tabulka 3: U1IE v host módu

Význam jednotlivých bitů je stejný jako výše.

Při připojení zařízení se vyvolá funkce obsluhy přerušení, uvnitř funkce se vykoná tato část:

```
void __attribute__((__interrupt__, no_auto_psv)) _USB1Interrupt( void )
{
    IFS5 &= 0xFFBF;

    if (U1IEbits.ATTACHIE && U1IRbits.ATTACHIF)
    {
        U1IEbits.ATTACHIE   = 0;
        U1IR                = USB_INTERRUPT_ATTACH;

        if (usbHostState == (STATE_DETACHED | SUBSTATE_WAIT_FOR_DEVICE))
        {
            usbOverrideHostState = STATE_ATTACHED;
        }
    }
{…}
}
```

Ukázka kódu 4: funkce přerušení od USB, připojení zařízení

V této části funkce vypnu další obsluhování přerušení vyvolané připojením zařízení a změním stav na zařízení připojeno. Pokud nastane odpojení zařízení, tak obdobnou funkcí opět nastavím původní hodnoty, aby program reagoval na nové připojení zařízení.

Nyní se v jádře programu dostanu ve stavovém automatu (dle obrázku 3) do větve zařízení připojeno.

```
case STATE_ATTACHED:
            switch (usbHostState & SUBSTATE_MASK)
            {
                case SUBSTATE_SETTLE:
                    switch (usbHostState & SUBSUBSTATE_MASK)
                    {
                        case SUBSUBSTATE_START_SETTLING_DELAY:
                            break;

                        case SUBSUBSTATE_WAIT_FOR_SETTLING:
                            break;

                        case SUBSUBSTATE_SETTLING_DONE:
                            break;
                    }
                    break;
```

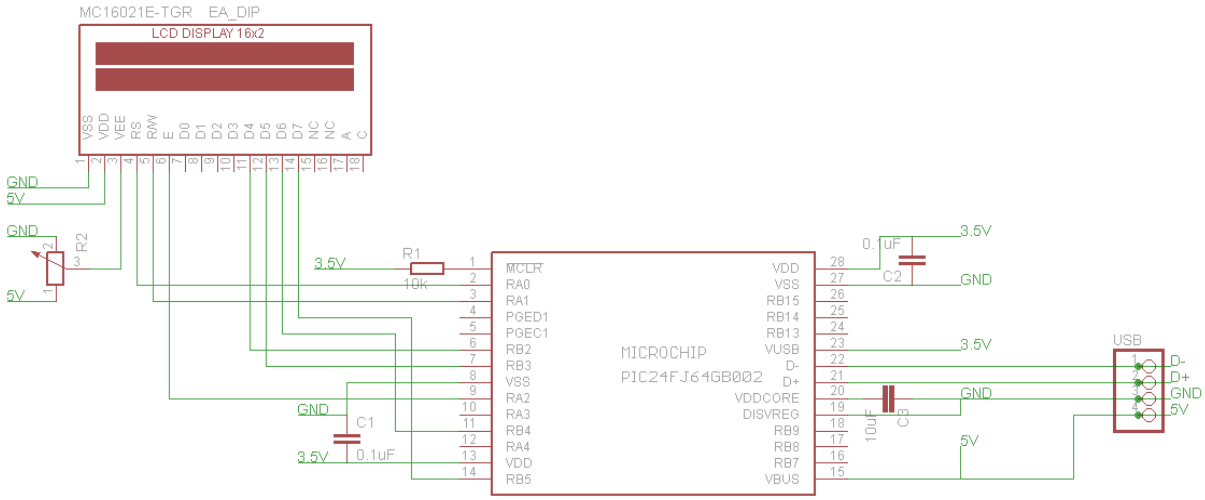
Ukázka z kódu 5: část stavového automatu

Stavy obsahují i podstavy, ve kterých se vykonávají kroky přesně definované výše v enumeraci USB. Ukázka celé implementace této enumerace by zabrala přibližně 6 stránek, proto zde uvádím pouze kostru automatu a první sadu podstavů. Po vykonání všech kroků se dostane program do stavu s nakonfigurovaným zařízením a je připraven k použití. To vše za předpokladu, že nenastala chyba. Mezi takové chyby se řadí například nepodporované zařízení, protože v mojí práci se zabývám pouze klávesnicí, jiné zařízení bude programem zamítnuto.

6 Celkové zapojení USB klávesnice a LCD displeje

Finální zapojení obsahuje LCD i USB komponenty. V této kapitole se budu věnovat finálnímu řešení zadání, a jak jsem vyřešil jednotlivé problémy. Mimo jiné se zaměřím na protokol HID pro klávesnici.

6.1 Schéma zapojení



Obrázek 5: Konečné schéma zapojení

6.2 Potřebná funkcionalita konečného řešení

Finální řešení musí umět ovládat displej a zobrazovat na něm výstup od klávesnice. K tomu, abych se dostal k danému konečnému stavu, musí program umět následující dílčí úlohy.

- Korektně nastavit mikroprocesor pro USB obsluhu a porty LCD
- Obsloužit připojení USB zařízení
- Rozpoznat připojené zařízení, konkrétně klávesnici
- Zpracovávat vstupní data na USB portu
- Přeformátovat data na ASCII symbol odpovídající stisknuté klávese
- Inicializovat připojené LCD displej
- Zapisovat na LCD displej

Zatím jsem rozepsal body o LCD displeji a připojení USB.

6.3 Rozpoznání připojeného zařízení na USB portu

Během vykonání výše popsané enumerace se kontroluje i typ připojeného zařízení. Pro můj případ použití klávesnice si musím ověřit, že je jedná o HID třídu s označením 3 se subtypypem, který je pro klávesnice 1.

6.4 Zpracování vstupních dat

Zpracování dat se vykonává na základě přerušení indikovaného klávesnicí, když je stisknuta nějaká klávesa. Na základě toho musí program přijmout input report. Tento input report obsahuje scancode stisknuté klávesy. Např. scancode pro klávesy *a* až *z* je 0x04 až 0x1D. Plný výpis je dostupný v dokumentu Keyboard Scan Code Specification, Appendix C [3].

```
void App_ProcessInputReport(void)
{
    BYTE  i;
    BYTE  data;

USBHostHID_ApiImportData(
      Appl_raw_report_buffer.ReportData,
      Appl_raw_report_buffer.ReportSize,
      Appl_BufferModifierKeys,
      &Appl_ModifierKeysDetails);

USBHostHID_ApiImportData(
      Appl_raw_report_buffer.ReportData,
      Appl_raw_report_buffer.ReportSize,
      Appl_BufferNormalKeys,
      &Appl_NormalKeysDetails);

for(i=0;i<(sizeof(Appl_BufferNormalKeys)/sizeof(Appl_BufferNormalKeys[0])
);i++)
    {
        if(Appl_BufferNormalKeys[i] != 0)
        {
            if(Appl_BufferNormalKeys[i] == HID_CAPS_LOCK_VAL)
            {
               CAPS_Lock_Pressed = !CAPS_Lock_Pressed;
               LED_Key_Pressed = TRUE;
               Appl_led_report_buffer.CAPS_LOCK = CAPS_Lock_Pressed;
            }else if(Appl_BufferNormalKeys[i] == HID_NUM_LOCK_VAL)
            {
                NUM_Lock_Pressed = !NUM_Lock_Pressed;
                LED_Key_Pressed = TRUE;
                Appl_led_report_buffer.NUM_LOCK = NUM_Lock_Pressed;
            }else
            {

if(!App_CompareKeyPressedPrevBuf(Appl_BufferNormalKeys[i]))
                {
                    data = App_HID2ASCII(Appl_BufferNormalKeys[i]);
```

```
                    LCD_Display_Routine(data,Appl_BufferNormalKeys[i]);
                }
            }
        }
        else
        {
            if(i==0)
            {
                HeldKeyCount = 0;
            }
            else
            {
                if(Appl_BufferNormalKeys[i-1] == HeldKey)
                {
                    if(HeldKeyCount < 3)
                    {
                        HeldKeyCount++;
                    }
                    else
                    {
                        data = App_HID2ASCII(HeldKey);
                        LCD_Display_Routine(data,HeldKey);
                    }
                }
                else
                {
                    HeldKeyCount = 0;
                    HeldKey = Appl_BufferNormalKeys[i-1];
                }
            }
            break;
        }
    }
    App_CopyToShadowBuffer();
    App_Clear_Data_Buffer();
}
```

Ukázka z kódu 6: Zpracování příchozích dat

6.5 Přeformátování na ASCII

Ve výše uvedeném kódu vidíme, že se volá funkce `App_HID2ASCII` (vyznačená tučně), tato funkce se stará o přeformátování scancode na ASCII znaky a její tělo vypadá takto:

```
BYTE App_HID2ASCII(BYTE a)
{
   BYTE AsciiVal;
   BYTE ShiftkeyStatus = 0;
   if((Appl_BufferModifierKeys[MODIFIER_LEFT_SHIFT]==1)||
      (Appl_BufferModifierKeys[MODIFIER_RIGHT_SHIFT] == 1))
   {
        ShiftkeyStatus = 1;
   }

   if(a>=0x1E && a<=0x27)
   {
       if(ShiftkeyStatus)
       {
            switch(a)
            {
                //zde sada case rozlišující znaky jako zavináč, atd.
            }

            return(AsciiVal);
       }
       else
       {
            if(a==0x27)
            {
                return(0x30);
            }
            else
            {
                return(a+0x13);
            }
       }
   }

   if((a>=0x59 && a<=0x61)&&(NUM_Lock_Pressed == 1))
   {
       return(a-0x28);
   }

   if((a==0x62) &&(NUM_Lock_Pressed == 1)) return(0x30);
```

```
   if(a>=0x04 && a<=0x1D)
   {
      if(((CAPS_Lock_Pressed== 1)&&
         ((Appl_BufferModifierKeys[MODIFIER_LEFT_SHIFT] == 0)&&
         (Appl_BufferModifierKeys[MODIFIER_RIGHT_SHIFT] == 0))||
         ((CAPS_Lock_Pressed == 0)&&
         ((Appl_BufferModifierKeys[MODIFIER_LEFT_SHIFT] == 1)||
         (Appl_BufferModifierKeys[MODIFIER_RIGHT_SHIFT] == 1))))
           return(a+0x3d); /* return capital */
      else
           return(a+0x5d); /* return small case */
   }

   if(a>=0x2D && a<=0x38)
   {
        switch(a)
        {
            //zde sada case rozlišující speciální znaky jako závorky atd.
            default:
                break;
        }
        return(AsciiVal);
   }
   return(0);
}
```

Ukázka z kódu 7: Konverze USB kódu na ASCII kód

6.6 Zobrazení na LCD displej

Po zpracování dat a přeformátování zbývá už jen výsledek, pokud je validní, zobrazit na LCD displej. K poslednímu rozpoznání kláves a zobrazení slouží tato funkce, kde se zpracovávají klávesy se speciální funkcí, jako například backspace, escape, šipky atd.:

```
void LCD_Display_Routine(BYTE data, BYTE HIDData)
{
    BYTE LineNum;
    BYTE CharPos;

    LineNum = ((currCharPos & 0x30) >> 4);
    CharPos = currCharPos & 0x0F;

      if((HIDData>=0x1E && HIDData<=0x27) ||
         (HIDData>=0x04 && HIDData<=0x1D) ||
```

```
          (HIDData>=0x2D && HIDData<=0x38) ||
          ((HIDData>=0x59 && HIDData<=0x62) &&
          (NUM_Lock_Pressed == 1)))
    {
        LcdPutch(data);
        currCharPos++;
    }
    else if(HIDData == 0x29)        // escape klávesa
    {…}
    else if (HIDData == 0x2C)       // mezerník
    {…}
    else if (HIDData == Symbol_Backspace) // backspace klávesa
    {…}
    else if((HIDData>=0x4F && HIDData<=0x52) ||

((HIDData==0x5C||HIDData==0x5E||HIDData==0x5A||HIDData==0x60)&&
           (NUM_Lock_Pressed == 0)))
    {
        switch(HIDData)
        {
            // zde sada case pro šipky
            default :
                break;
        }

    }
    //zde kontrola pozice na displeji a pohyb/mazání
}
```

Ukázka z kódu 8: Výstup na LCD displej

7 Vyhodnocení spolehlivosti řešení

V průběhu implementace zapojení USB jsem se potkal s několika problémy. Jedním z nich byl, že některé klávesnice slouží zároveň jako USB rozbočovač, to znamená, že klávesnice je zapojená až do rozbočovačem. Tento typ klávesnic je programem zamítnut, protože rozbočovače nejsou podporovány. Jako další jsou speciální klávesy, kterou jsou závislé na daném modelu, v řešení se zabývám pouze standardním rozložením kláves. A poslední problém se kterým jsem se setkal, je, že input report klávesnice není přesně dán, toto je problém převážně v implementaci typu jako je např. tato práce, kde nemám prostor pro implementace dle různých výrobců a typů klávesnic.

8 Závěr

Cílem práce bylo připojit klávesnici přes USB port k mikroprocesoru mojí volby a stisknuté klávesy zobrazovat na LCD displej. V průběhu práce jsem se musel seznámit s vybraným mikroprocesorem, LCD displejem, USB protokolem, HID protokolem a celkovou komunikací přes USB port. Tyto body jsem postupně probíral v kapitolách. Výsledkem bylo sestavení programů z komponent, které byly dostupné z aplikačních knihoven Microchip. Výjimkou byl LCD displej, pro který jsem program vytvářel dle znalostí ze studia na VŠB-TUO. Případný další vývoj tohoto projektu je vhodný ve směru univerzálnosti a nezávislosti na typu použité klávesnice. Tato práce souvisí s prácí Patrika Slučiaka s názvem „LCD textový terminál“.

Reference

[1] COMPAQ COMPUTER CORPORATION A KOL. *Universal Serial Bus Revision 2.0 specification: usb_20.pdf* [online]. 27. 4. 2000 [cit. 2014-05-05].
Dostupné z: http://www.usb.org/developers/docs/usb20_docs/usb_20_042814.zip

[2] USB IMPLEMENTERS' FORUM. *Device Class Definition for Human Interface Devices (HID)* [online]. 27. 6. 2001 [cit. 2014-05-05].
Dostupné z: http://www.usb.org/developers/devclass_docs/HID1_11.pdf

[3] MICROSOFT. *Keyboard Scan Code Specification* [online]. 16. 3. 2000 [cit. 2014-05-05].
Dostupné z:http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/scancode.doc

[4] MICROCHIP. *MPLAB® C30 C COMPILER USER'S GUIDE* [online]. 8. 12. 2006 [cit. 2014-05-05]. Dostupné z:http://ww1.microchip.com/downloads/en/devicedoc/c30_users_guide_51284f.pdf

[5] MICROCHIP. *PIC24FJ64GB004 Family Data Sheet* [online]. 15. 7. 2010 [cit. 2014-05-05].
Dostupné z:http://ww1.microchip.com/downloads/en/DeviceDoc/39940d.pdf

[6] EVERBOUQUET/WAYTON. *General specifications for character LCD module* [online]. [cit. 2014-05-05]. Dostupné z: http://www.everbouquet.com.tw/Gener.htm

[7] MICROCHIP. *Applications library* [online]. [cit. 2014-05-05].
Dostupné z:http://ww1.microchip.com/downloads/en/softwarelibrary/microchip-libraries-for-applications-v2013-06-15-windows-installer.exe